Abstract

The amount of software in many systems increases exponentially. This increase impacts the reliability of these systems. In the source code of software many hidden faults are present. These hidden faults can transform into errors during the system life cycle, due to changes in the system itself or in the context of the system.

We will discuss the current trends and potential directions for future solutions of an increasing reliability problem.
Introduction
  + ESI
  + Speaker

Exploration
  + trends
  + consequences for reliability
  + potential solutions

Research Projects
  + Trader
  + Tangram
  + Ideals
  + Boderc

Conclusion
Embedded Systems Institute (ESI)

**Mission**
To advance industrial innovation and academic excellence in embedded systems engineering (ESE).

**Vision**
ESI and its partners create and apply world-class ESE methods.

7 Founders:
- Industry (Philips, ASML, Océ)
- Universities (Twente, Delft, Eindhoven)
- Knowledge Institutes (TNO)

- television
- waferstepper
- printer
- GSM
- cardio X-ray system
Embedded Systems Engineering

- performance
- reliability
- evolvability

in relation to all other system qualities: security, power, cost, size, et cetera
Who is Gerrit Muller?

CT OSS NM X-ray

1980 1990 2000

Ph: Philips
AM: ASML

Philips Medical

industrial experience

time pressure pragmatics sales

cost constraints products

lots of people

research

reflection evidence exposure education

Buskerud

EasyVision

manager system engineering

Gaudí

rel: Reliability of Software Intensive Systems

version: 0.2

June 21, 2020

FIESAwhoIsGerrit
Exploration of Reliability

Introduction
  + ESI
  + Speaker

Exploration
  + trends
  + consequences for reliability
  + potential solutions

Research Projects
  + Trader
  + Tangram
  + Ideals
  + Boderc

Conclusion
Trends in Embedded Systems

- How to survive in innovative domains?
- Fast moving market
- Fast moving technology
- Complex value chains
- Increased integration

Reliability of Software Intensive Systems
Gerrit Muller
version: 0.2
June 21, 2020
ECMAproblem
Increased Team Size

Reliability of Software Intensive Systems

version: 0.2
June 21, 2020
DYOFteamsize
Number of Faults Proportional With Code Size

Based on average 3 errors/kloc
The Hard Reset Syndrome: Power Down Needed!

Hard Reset Required:

Cell Phone
Television
PC
Beamer
Car
Coffee Machine
DVD player
Airbus

Pilot announces a flight delay, due to computer problems. A complete reset is required. The flight entertainment system also show a reset: a complete Linux boot. This reboot hangs: server xxx not found
How to Make SW Intensive Systems Reliable

- within cost & time
- improve reliability
- without performance penalty
- with increasing requirements
- increasing # suppliers # sites, et cetera

more/longer testing
more procedures
reduce functionality
improve way-of-working
reduce code
more standardization
improve recovery
improve detection
improve robustness

more formality
more automation
early integration
shared understanding
more feedback
tolerant interfaces
robust patterns
Reliability Research

Introduction
  + ESI
  + Speaker

Exploration
  + trends
  + consequences for reliability
  + potential solutions

Conclusion

Research Projects
  + Trader
  + Tangram
  + Ideals
  + Boderc
**Objective:**
Develop method & tools to ensure reliable consumer electronics products

**Research agenda:**
System Reliability

**Domain:**
Digital TV

**CIP:**
Philips Semiconductors

**Partners:**
Philips CE, Tass, and Research

**Timeline:**

**Industrial Relevance**

Poor reliability has severe business impact

- Customer expectation of TV reliability is high
  - Little tolerance for technical problems
- 100% fault-free design is not achievable
- High volume market implies high risks if reliability problems occur
  - Low product margin leaves no buffer for service costs
  - Service center costs multiplied by number of complaints
  - Market share reduction likely, i.e. customers buy another brand
- (On) Time to market is critical
  - Missing fixed shipping gates costs millions of dollars

**User Perceived Reliability**

**Objective**
- Determine the user-perceived severity of a product failure mode

**Methodology**
- Create a model considering relevant factors
  - User-perceived loss of functionality
  - User-perceived reproducibility
  - Failure-frequency
  - Work-around difficulty
  - User-group characteristics
  - Failure characteristics
  - ...

<table>
<thead>
<tr>
<th>Aspects (all depend on user)</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Function importance</td>
<td>4</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>Frequency</td>
<td>4</td>
<td>1</td>
<td>5</td>
</tr>
<tr>
<td>Reproducibility</td>
<td>3</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>Solvability</td>
<td>3</td>
<td>2</td>
<td>5</td>
</tr>
<tr>
<td>Loss of Function / time / behavior</td>
<td>4</td>
<td>5</td>
<td>0</td>
</tr>
<tr>
<td>.....</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Total:</td>
<td>576</td>
<td>160</td>
<td>0</td>
</tr>
</tbody>
</table>

- Validate model

- Evaluate and suggest system failure recovery strategies
  - The recovery strategy may not annoy the user even more!

**Trader Domain Trends**

TV complexity increase follows the PC world

from “Display movies over antenna”

to “Display anything over everything”

**Broadcast**
ATSC, DVB, ISDB, Analog
Terrestrial, Satellite, Cable

**Connectivity**
Cameras, JPEG, flash cards, HDD, MP3, Web browser, Ethernet, USB, etc.

**Overview**

**Trader**
Television Related Architecture and Design to Enhance Reliability

Objective:
Develop method & tools to ensure reliable consumer electronics products

**Research agenda:**
System Reliability

**Domain:**
Digital TV

**CIP:**
Philips Semiconductors

**Partners:**
Philips CE, Tass, and Research

**Timeline:**

**User Perceived Reliability**

**Objective**
- Determine the user-perceived severity of a product failure mode

**Methodology**
- Create a model considering relevant factors
  - User-perceived loss of functionality
  - User-perceived reproducibility
  - Failure-frequency
  - Work-around difficulty
  - User-group characteristics
  - Failure characteristics
  - ....

<table>
<thead>
<tr>
<th>Aspects (all depend on user)</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Function importance</td>
<td>4</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>Frequency</td>
<td>4</td>
<td>1</td>
<td>5</td>
</tr>
<tr>
<td>Reproducibility</td>
<td>3</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>Solvability</td>
<td>3</td>
<td>2</td>
<td>5</td>
</tr>
<tr>
<td>Loss of Function / time / behavior</td>
<td>4</td>
<td>5</td>
<td>0</td>
</tr>
<tr>
<td>.....</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Total:</td>
<td>576</td>
<td>160</td>
<td>0</td>
</tr>
</tbody>
</table>

- Validate model

- Evaluate and suggest system failure recovery strategies
  - The recovery strategy may not annoy the user even more!
Increasing Code Size in Televisions

From: COPA tutorial, Rob van Ommering

1965: 1 kB
1979: 1 kB
2000: 2 MB
1990: 64 kB

Moore's law
Research: System Awareness to Improve Reliability

User input → system → system output

- Awareness
- Monitor
- Correction

Customer expectations
System failure model
Research: Code Analysis to Improve Reliability

Expose product weaknesses at design time

- Source code analysis
  - Identify hotspots in code
  - Consider impact to user-visible behavior: prioritize warnings

- Software architecture reliability analysis
  - Techniques to identify failure-prone components
  - Evaluation of architectural alternatives and trade-offs
Quality Degradation Caused by Shit Propagation

- needed code
- repair code
- new needed code
  - code not relevant for new function
  - repair code
  - bad code
- new bad code

Bad code

Copy paste modify

Reliability of Software Intensive Systems
19 Gerrit Muller

version: 0.2
June 21, 2020
BLOAT shitPropagation
Example of Shit Propagation

Class Old:
    capacity = startCapacity
    values = int(capacity)
    size = 0

    def insert(val):
        values[size]=val
        size+=1
        if size>capacity:
            capacity*=2
            relocate(values, capacity)

Class New:
    capacity = 1
    values = int(capacity)
    size = 0

    def insert(val):
        values[size]=val
        size+=1
        capacity+=1
        relocate(values, capacity)

Class DoubleNew:
    capacity = 1
    values = int(capacity)
    size = 0

    def insert(val):
        values[size]=val
        size+=1
        capacity+=1
        relocate(values, capacity)

    def insertBlock(v,len):
        for i=1 to len:
            insert(v[i])
Example Tangram Project

Integration & Test

Early Integration

- Incremental
- Model Based
- Model ⇔ Realization
- Infrastructure
- NDDS, Matlab, Chi

Results:
- Early Model Based Integration and Testing in AGILE
- Problem found in S&CoMo before review

Test Selection & Sequencing

Model Based Testing

Reliability of Software Intensive Systems

version: 0.2
June 21, 2020
RSISOBoro
Challenge: Exponential Increase

- Performance
- Imaging
- Overlay
- Productivity
- Reliability
- Robustness
- Innovation
- Feedback and Adjustments
- HW and SW Components
- Complexity

Reliability of Software Intensive Systems

version: 0.2
June 21, 2020
ASMLproblemIRC
4 Views on a Waferstepper

subsystems

control hierarchy

kinematic

physicsoptics

Reliability of Software Intensive Systems
24 Gerrit Muller
Research: Test Strategy

<table>
<thead>
<tr>
<th>S</th>
<th>T</th>
<th>Test t0</th>
<th>Test t1</th>
<th>Test t2</th>
<th>Test t3</th>
<th>Test t4</th>
<th>Test t5</th>
<th>P</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fault state 1</td>
<td></td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>10%</td>
</tr>
<tr>
<td>Fault state 2</td>
<td></td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>10%</td>
</tr>
<tr>
<td>Fault state 3</td>
<td></td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>10%</td>
</tr>
<tr>
<td>Fault state 4</td>
<td></td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>10%</td>
</tr>
<tr>
<td>Fault state 5</td>
<td></td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>10%</td>
</tr>
<tr>
<td>C</td>
<td></td>
<td>3</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td></td>
</tr>
</tbody>
</table>

Dynamic simulation of integration & test approach:
1. Create model (modules, interfaces, faults, tests)
2. Execute model at any time
3. Balancing based on time, cost and remaining risk.

Balancing functionality, quality and time/cost
(over 20% reduction of integration & test time)
Example Ideals Project

code size \(\overset{\text{reduce by}}{\rightarrow}\) cross cutting concerns

Results: Parameter Check

• Automatic replacement of current parameter checking and tracing idiom with specific aspect oriented idiom.

• Code size reduction of 80% for parameter checking idiom (7% reduction of total module)

• Improved locality
Example Boderc Project

31x5F  2050  2090
Boderc Goal

Boderc goal =

A specific methodology

based on modeling

and analyze, discuss, document, and communicate

throughput, quality

multi-disciplinary

system performance

within industrial constraints and restricted design space

people, process, project duration, and cost

power computing response time
Shared Understanding by Modeling

- System
- Multi-disciplinary
- Mono-disciplinary
- Formula based
- Back-of-the-envelop
- Executable

Reliability of Software Intensive Systems
29 Gerrit Muller

version: 0.2
June 21, 2020
ESICpyramidModels
6. Kinematic modeling

small, simple, goal-driven models

shorter cycle time, less cycles

shorter product creation lead time
Coverage of Reliability by ESI Projects

within cost&time
improve reliability
without performance penalty
with increasing requirements
increasing #suppliers #sites, et cetera

more/longer testing
more procedures
reduce functionality
improve way-of-working
reduce code
more standardization
improve recovery
improve detection
improve robustness

more formality
more automation
early integration
shared understanding
more feedback
tolerant interfaces
robust patterns

Ideals
Tangram
Boderc
Trader
Trader
Towards a Conclusion, Some more Trends

Introduction
  + ESI
  + Speaker

Exploration
  + trends
  + consequences for reliability
  + potential solutions

Research Projects
  + Trader
  + Tangram
  + Ideals
  + Boderc

Conclusion
Applications depend on chain of systems

users

infotainment appliance
watch video
browse photo's
calendar
and much more...

Home Server Network Providers Service Providers Content Providers
Interoperability: systems get connected at all levels

Reliability of Software Intensive Systems

version: 0.2
June 21, 2020
DYOFscopeOfInteroperability

Gerrit Muller

34
Multi-dimensional interoperability

integrating *multiple* applications
  clinical analysis
  clinical support
  administrative
  financial
  workflow

in *multiple* languages
  cultures
  USA, UK,
  China, India,
  Japan, Korea
  France, Germany
  Italy, Mexico

based on *multiple* media, networks
  DVD+RW
  memory stick
  memory cards
  bluetooth
  11a/b/g
  UMTS

and *multiple* standards
  Dicom
  HL7
  XML

and *multiple* releases
  R5
  R6.2
  R7.1

delivered by *multiple* vendors
  Philips
  GE
  Siemens
Interoperability Trends and Research Challenges

<table>
<thead>
<tr>
<th>trends</th>
<th>(partial) solutions</th>
</tr>
</thead>
<tbody>
<tr>
<td>market dynamics</td>
<td>standards</td>
</tr>
<tr>
<td>globalization</td>
<td>economical interests</td>
</tr>
<tr>
<td>hype waves</td>
<td>latency due to slow process</td>
</tr>
<tr>
<td>Moore's law</td>
<td>most fundamental solution</td>
</tr>
<tr>
<td>interoperability</td>
<td>design patterns</td>
</tr>
<tr>
<td>emerging behavior</td>
<td>in system feedback</td>
</tr>
<tr>
<td>future vs legacy</td>
<td>human-in-the-loop feedback</td>
</tr>
<tr>
<td>heterogeneous vendors</td>
<td>semantic understanding</td>
</tr>
<tr>
<td>dynamics (continually changing)</td>
<td>containment</td>
</tr>
<tr>
<td>reliability</td>
<td>gateway</td>
</tr>
<tr>
<td>complexity</td>
<td>new research challenges!</td>
</tr>
<tr>
<td>heterogeneity</td>
<td></td>
</tr>
<tr>
<td>dynamics</td>
<td></td>
</tr>
<tr>
<td>#engineers involved</td>
<td></td>
</tr>
</tbody>
</table>

version: 0.2
June 21, 2020
RSISinteroperability
Conclusion

Based on average 3 errors/kloc
less errors/kloc
Based on average


1000
10Mloc
100
1Mloc
10
100 kloc

Reliability of Software Intensive Systems
37 Gerrit Muller

version: 0.2
June 21, 2020
RSISconclusion